Parser plugin

What's that language you're writing this plugin for?

This is a server-side scripting language called "parser". It was developed by Art.Lebedev Studio and you can read about it on it's site.

It's syntax is very complicated for the plugin writer and maybe less complicated for a person who code scripts in this language: for instance you can refer to a variable named "chance" like this:

1. $chance
2. $[chance]
3. ${chance}

Or with a variable temp ($temp[chance] - this is the assignment operator - assign string "chance" to variable "temp"):

1. $$temp
2. $[$temp]

So it's quite difficult to implement all this, though i'm not crazy about doing it - i think that couple of common forms is of real need and the others have a true low priority.

Who would use your plugin?

The truth is that i personally need it since i work with this "parser" language. It helps me to develop things quickly and with less errors. So it would be useful for me or other people who deal with "parser".

And by no means i want you to start writing code in "parser" - you can if you want to but it's on your own risk; you've been warned.

Detail description

By now the plugin is capable of:

- highlight syntax
- build a psi tree (file, class, method definition, object definition, object reference, method reference)
- find references from classes, objects and methods
- complete local object, methods, classes in index and object instances which have a value of some classes' constructor call
- match braces
- fold methods and imports
- collect classes into class index

Open questions

Please be sure to share your point of view - it is of great importance to me.

1. The role of flex-generated lexer: should i include different states in it to parse grammatical structures or it should be pretty simple and all the grammar should be implemented at the PsiParser level?

2. How to handle classes which are accessible but not directly included into one particular file? I've noticed the similar problem while working with the xpath/xslt plugin: imagine you've got one xslt with two imports, so that in the second one you may refer to a variable declared in the first one; but it's very unclear how to resolve this reference.